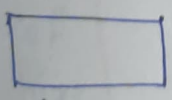
structures

→ It can be defined as collection of data members that are related data members under one name, and those data members may be of similar type, may be of dissimilar type.

→ It is defined as collection of the dissimilar data items under one name i.e grouping the data items

→ structure is used for defining user defined data types apart from the primitive data types

→ Using primitive data types like int, float, char, long etc, we can define our own data type depending on our requirement.



Structure definition

```
struct Rectangle
{
    int length;
    int breadth;
};
```

→ If you just define memory will not be consumed.

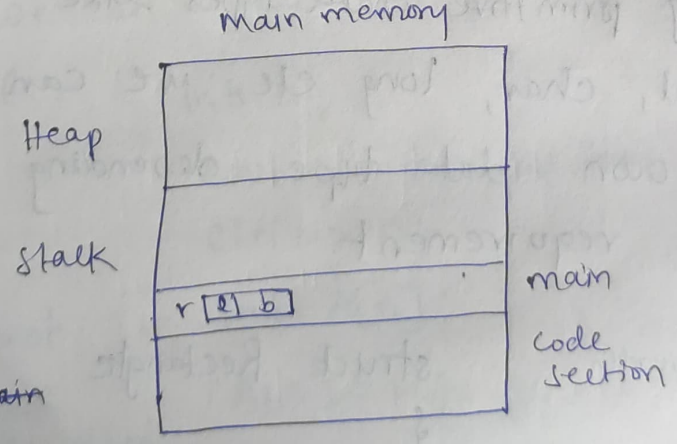
object (or) variable

→ if you create a variable using that, then memory is consumed.

→ In above case, if you consider 4 bytes for integer. It could consume 8 bytes for that structure.

```
int main()
{
    struct Rectangle r;  // declaration
// dec + init  struct Rectangle r = {10, 5};
    r.length = 15;
    printf("Area = %d", r.length * r.breadth);
}
```

o/p: 75



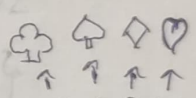
→ you can access, modify data members of structures using dot operator.

More examples on structure

→
```
struct student
{
    int roll;
    char name[25];
};
struct student s;
s.roll = 21;
s.name = "Amar";
```

This is for a 1 student, we can create many students data using this.

→
```
struct card
{
    int face;
    int shape;
    int color;
};
int main() {
struct card deck[52] = {{1,0,0},{2,0,0},
                        {1,1,0}, ....};
printf("%d", deck[0].face); // 1
printf("%d", deck[1]); → Try this
}
```



shape - 0, 1, 2, 3

color - 0, 1

Black Red

face - 1 - 13

A-1, 2-10, 11-J, 12-Q, 13-K

* Array of structure

practise · structures

→ we can declare variable of struct type.

Method 1:

```
struct Rectangle
{
    int length;
    int breadth;
} r1, r2, r3;
```

(or)

Method 2:

```
struct Rectangle
{
    int length;
    int breadth;
};
struct Rectangle r1;
int main ()
{
    cout << "Hello world";
}
```

* while declaring in these two methods it is available to all functions because it's declared outside main function.

Method 3:

Declaring in Main function.

→ Inside structures, padding of memory is done means it will take extra memory.

for Example:

```
struct Rectangle
{
    int length;      - 4 bytes
    int breadth;     - 4 bytes
    char x;          - 1 bytes
};
```

↓

* But it will take 4 bytes instead of 1

→ we can find size of structure using sizeof operator.

* cout << sizeof(r1) << endl;